electronics

MDPI

Article

Polynomial Algorithm for Minimal (1,2)-Dominating Set in Networks

Joanna Raczek

Department of Algorithms and Systems Modelling, Faculty of Electronics, Telecommunications and Informatics, Gdańsk University of Technology, Narutowicza 11/12, 80-233 Gdańsk, Poland; joanna.raczek@pg.edu.pl

Abstract: Dominating sets find application in a variety of networks. A subset of nodes D is a (1,2)-dominating set in a graph $G = (V, E)$ if every node not in D is adjacent to a node in D and is also at most a distance of 2 to another node from D. In networks, (1,2)-dominating sets have a higher fault tolerance and provide a higher reliability of services in case of failure. However, finding such the smallest set is NP-hard. In this paper, we propose a polynomial time algorithm finding a minimal (1,2)-dominating set, `Minimal_12_Set`. We test the proposed algorithm in network models such as trees, geometric random graphs, random graphs and cubic graphs, and we show that the sets of nodes returned by the `Minimal_12_Set` are in general smaller than sets consisting of nodes chosen randomly.

Keywords: computational complexity; algorithms; domination number; (1,2)-domination number; networks

Citation: Raczek, J. Polynomial Algorithm for Minimal (1,2)-Dominating Set in Networks. *Electronics* **2022**, *11*, 300. https://doi.org/10.3390/electronics11030300

Academic Editors: Juan M. Corchado, Stefanos Kollias and Javid Taheri

Received: 30 December 2021
Accepted: 17 January 2022
Published: 19 January 2022

1. Introduction

In networks, some resources (possibly limited in number) should be available immediately and directly. Thus, some nodes can function as special nodes, for example, servers, radio broadcasting stations, schools or hospitals in networks of streets, and towns or countries. Then these special nodes form a dominating set. Since we want to minimize the total costs of devices, facilities or buildings, we are interested in dominating sets of small cardinality. The domination number is the number of the smallest possible number of these special nodes.

However, some networks demand a higher reliability, so in the case of one server's failure, a different one can take over necessary tasks. In this paper, we focus on a situation when a spare special node does not need to be a direct neighbour of an ordinary node—it might be at distance of 1 or 2. If a spare special node is at distance 2, then the communication might be performed at a lower speed or only the most demanding tasks may be assured by the spare server. An analogous situation takes place when the poor signal coverage (because of a failure) makes it impossible to make standard connections via a mobile phone network. However, in most situations we can call emergency numbers. During pandemics we want to be sure that in case of overcrowding of the nearest hospital, there is a spare one at most two distances further. Similar situation takes place when we want to manage the location of maternity wards. To fulfil these conditions, the special nodes need to form a (1,2)-set, also known as a secondary dominating set. Of course, determining the smallest (1,2)-set is most desirable.

The execution time of a polynomial–time algorithm can be bounded by from above by a polynomial on the size of the input. For this reason such algorithms have practical use in networks. The Information System on Graph Classes and their Inclusions [1] shows what we know about computational complexity of graph theory problems. It can be noted that for most types of dominating set problems determining the smallest set of the desired property is NP-hard in general graphs. This means that polynomial algorithms solving these problems are not known (for more information about NP-hard problems and polynomial algorithms see [2]). On the other hand, polynomial–time algorithms

exist for specific graph classes. For example, finding the minimum domination number is polynomial in trees, block graphs, interval graphs, cographs and permutation graphs (sample algorithms with examples can be found in [3,4]).

For wider and more general graph classes approach on the other way can be made. Approximate polynomial algorithms are developed, however they do not ensure that the returned results are best possible, that is of smallest cardinality and dominating at the same time. Since (1,2)-dominating sets have potential for many applications in real-life situations and up to now there is a little known about algorithms finding these sets, in our paper we propose a polynomial time algorithm that finds a minimal (1,2)-dominating set in any graph. Such a set can be regarded as good heuristic of the minimum (the smallest) (1,2)-dominating set (see Section 3.1 for more details on minimum and minimal sets and Section 4.2 for results for trees).

This paper is organized as follows. In Section 3, we present the formal definitions used in a model of a network and we state the difference between minimum and minimal sets. Next we present algorithms `PowerOfNodes` and `Minimal_12_Set`. In Section 4, we analyze the performance of the algorithms in geometric random graphs in which nodes are placed randomly on a unit square and two nodes are adjacent if and only if the distance between them in Euclidean space is at most given threshold. This is done to predict the results given by the algorithm in a real network situation. Moreover, we compare the results returned by the algorithm with exact values of (1,2)-domination number in tree graphs. This helps us check how much the algorithm is better than simply choosing nodes randomly. At last, we investigate performance of the algorithm in random graphs and cubic graphs. In Section 6, we give conclusion of this paper and discuss future further works.

2. Related Works

A review of the literature shows that domination parameters can be used in many applications, ranging from diverse sensor networks [5] to vehicular ad hoc communications [6,7], DNA sequencing [8], safety and reliability in transportation [9], disaster rescue operations [10], and many others. In all these works graphs are used to model the network. In case of Wireless Sensor Networks, mainly dominating sets, connected dominating sets and weakly connected dominating sets are studied. The connected dominating set is a model of a virtual backbone, which helps achieve efficient broadcasting in these networks [11,12]. Connected domination has also applications to ad hoc networks and there are a few papers on this topic—see, for example [6,13].

Until now, there have not been many papers on $(1,2)$ domination. The concept of (1,2)-dominating sets in graphs was introduced by Hedetniemi et al. [14] as secondary domination. They also generalized this notion to (1,k)-domination, where k is a positive integer. In [15], the authors study (1,2)-domination number in tournaments, and in [16] graphs with the (1,2)-domination number close to their orders are characterized. An independent version of (1,2)-domination in graphs is studied in [17]. In particular the authors investigate graphs which posses a (1,2)-dominating set such that no two nodes of the set are adjacent.

The computational complexity of determining the smallest (1,2)-dominating sets in graphs and a polynomial time algorithm finding the (1,2)-domination number in trees are presented in [18]. In particular, the authors prove that determining the (1,2)-dominating number is NP complete, even for split graphs and even for bipartite graphs and many other graph classes. In the same paper, the author prove that if a graph does not have nodes of degree one nor triangles (three nodes such that each two of them are adjacent), then the (1,2)-domination number is equal to the domination number. The problem is that determining the domination number is also NP-hard. Furthermore, in the case of ad hoc networks, we may never be sure whether the network contains triangles of nodes of degree one without constant checking the structure of the network.

The main differences between this contribution and published works is that the published works do not consider approximate algorithms that find the (1,2)-domination

number for general graphs. Even though there are some algorithms for dominating sets and connected dominating sets (for example a self-stabilizing algorithm finding a minimal dominating set is presented in [19] and an exact algorithm for the domination number is given in [20], however it runs in exponential time) they can not be applied to (1,2)-domination. By the definition, (1,2)-domination number is never bigger than domination number. Similarly, if the connected domination number of a graph is equal to 1, then the (1,2)-domination number is equal 2, however in all other cases the (1,2)-domination number is not greater than the connected domination number. Moreover, the difference between these two dominating numbers may be big relative to the number of nodes (for a path on 100 nodes the difference between these two domination parameters is 64). Hence, we conclude that algorithms for connected dominating sets do not apply here.

Since determining the exact value of (1,2)-domination number is known to be NP-hard for chordal bipartite graphs, C_4-free graphs, maximum degree 4 graphs, partial grid graphs and planar graphs (see [18]), in this paper we propose an algorithm that finds a minimal (1,2)-dominating set in an arbitrary graph in a polynomial time. By analyzing neighbourhoods of nodes, our algorithm aims to construct a minimal (1,2)-dominating sets of cardinalities close to the (1,2)-domination number.

3. Materials and Methods

In this section, we explain all necessary definitions and also indicate the difference between the minimum and minimal (1,2)-dominating sets. Next, we present the two main algorithms of this work.

3.1. Model of Network

In a formal way, let an undirected graph $G = (V, E)$ be a model of a network. Namely, let V be the set of nodes and E the set of two element subsets of V, called links. If $\{u, v\} \in E(G)$ is a link, then we will write uv for short. If $v \in V(G)$, then $N_1(v) = \{u \in V(G) : vu \in E(G)\}$ is the set of all neighbours of v and $N_2(v)$ is the set of nodes in G at distance exactly 2 from v, that is $N_2(G) = \{u \in V(G) : vz, zu \in E(G)$ for some $z \in V(G)$ and $uv \notin E(G)\}$. The number of elements in $N_1(v)$ is called the degree of v.

A set $D \subseteq V(G)$ is a dominating set in of graph $G = (V, E)$ if every node not in D is adjacent to a node in D. A set $D \subseteq V(G)$ is a (1,2)-dominating set in G if every node not in D is adjacent to a node in D and is also at distance at most 2 to another node from D. Hence each (1,2)-dominating set is also a dominating set, but not vice versa. The (1,2)-domination number, denoted by $\gamma_{(1,2)}(G)$, is the cardinality of a smallest (1,2)-dominating set of G.

We say that a node v is (1,2)-dominated by a subset $D \subseteq V(G)$ if v is adjacent to a node belonging to D and there is another node in D at distance at most 2 from v.

A (1,2)-dominating set is a minimal (1,2)-dominating set of a graph G if no proper subset of D is a (1,2)-dominating set of a G. The following example clarifies the difference between minimum and minimal (1,2)-dominating sets.

For example, in Figure 1 the sets $\{A, B, D, H\}, \{A, C, F, H\}, \{B, E, G\}$ and $\{B, E, H\}$ are examples of minimal (1,2)-dominating sets, but only the sets with three nodes are minimum (1,2)-dominating sets. The set $\{A, B, D, G\}$ is not a (1,2)-dominating set, as although H has a neighbour in this set (namely G), no other node of the set is at distance at most 2 from H. On the other hand, a set $\{B, E, G, H\}$ is a (1,2)-dominating set, but not minimal, because its proper subset, $\{B, E, G\}$ is also a (1,2)-dominating set.

In some cases, in graph G the difference between the (1,2)-domination number and a cardinality of a minimal (1,2)-dominating set can be large in relation to the number of nodes of G. For example, let us consider a star $K_{1,t}$ with t nodes of degree 1. Then $\gamma_{(1,2)}(K_{1,t}) = 2$, while the set of all nodes of degree 1 form the biggest minimal (1,2)-dominating set. The algorithm presented in this paper tries to avoid constructing such a biggest minimal (1,2)-dominating sets. In particular, for stars it always returns a solution with two nodes, which is the best in terms of the number of nodes.

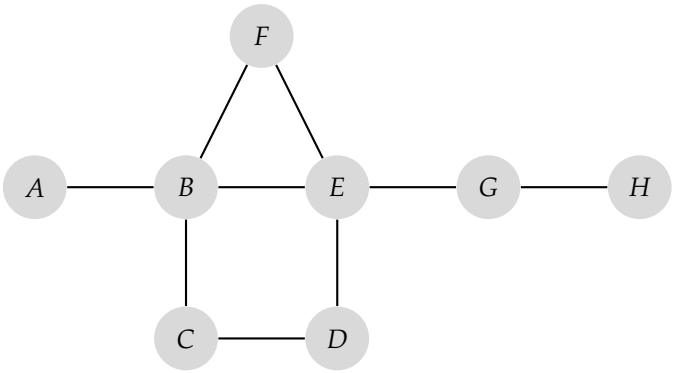

Figure 1. Example of a graph.

3.2. Algorithm `PowerOfNodes`

The algorithm presented here finds a minimal (1,2)-dominating set D in G. In the beginning, D is an empty set. In each main step of the algorithm, a new node is added to D until each node in $V - D$ has a neighbour in D as well as is at distance at most 2 to another node in D.

Each node has three local variables: *code*, *dom* and *pow*. These variables show the state of each node and its neighbours.

dom If a node v is chosen to be in D, then $v.dom$ is equal to 1. Otherwise, $v.dom = 0$.

code If $v.dom = 1$, then $v.code = 30$.

If $v.dom = 0$ and if additionally

- For each node x of $N_1(v) \cup N_2(v)$ is $x.dom = 0$, then $v.code = 0$.
- There is exactly one node x in $N_1(v)$ such that $x.dom = 1$ and for each node x of $N_2(v)$ is $x.dom = 0$, then $v.code = 10$.
- There is at least one node x in $N_2(v)$, such that $x.dom = 1$ and for each node x of $N_1(v)$ is $x.dom = 0$, then $v.code = 5$.
- v is (1,2)-dominated by nodes in D, then $15 \leq v.code \leq 20$.

pow This parameter shows the gain of adding v to D and its value depends on two algorithm parameters: a_1 and a_2. If v is already chosen to the (1,2)-dominating set, then $v.pow = 0$. Otherwise, $v.pow = n_0 + a_1 \cdot n_1 + a_2 \cdot n_2$, where

- $n_0 = 1$ if v is not (1,2)-dominated and otherwise $n_0 = 0$.
- n_1 is the number of nodes in $N_1(v)$ that do not have a neighbour in D.
- n_2 is the number of nodes in $N_2(v)$ that have exactly one neighbour in D but are not (1,2)-dominated by D or do not have any neighbours belonging to D within distance 2.

By assigning different values to parameters a_1, a_2 we can obtain different (1,2)-dominating set D.

To introduce an algorithm for finding a minimal (1,2)-dominating set in arbitrary graphs, we first introduce a function `PowerOfNodes` which has four parameters: a graph G, a node $u \in V(G)$ and two positive numbers: a_1, a_2. The function determines the powers of u, each neighbour of u and each node at distance 2 from u.

Algorithm `PowerOfNodes` Analysis

Let v be a node in $N_1(u) \cup N_2(u) \cup \{u\}$. The Algorithm 1 first checks (if…else lines 3 and 14–15) whether v is in D (then the power of v is always 0) or not. If not and if v is not (1,2)-dominated, then v gets power at least 1 (lines 5–6). This assures a proper work of the Algorithm 1 in case when, for example, v is not connected to any other node.

Algorithm 1: `PowerOfNodes`

Data: A graph G, a node u and parameters a_1, a_2
Result: G with determined power of nodes in $N_1(u) \cup N_2(u) \cup \{u\}$

```
1  Function PowerOfNodes(G, u, a1, a2):
2      for v ∈ N1(u) ∪ N2(u) ∪ {u} do
3          if v.dom = 0 then
4              m = 0;
5              if v.code ∈ {0, 5, 10} then
6                  m ← 1
7              for x ∈ N2(v) do
8                  if x.code ∈ {0, 10} then
9                      m ← m + a2;
10             for x ∈ N1(v) do
11                 if x.code < 11 then
12                     m ← m + a1;
13             v.pow ← m;
14         else
15             v.pow ← 0
16     return G
```

For each node not in D belonging to $N_2(v)$ with code 0 or 10, we add a_2 to $v.pow$ (lines 7–9). Similarly, for each node not in D belonging to $N_1(v)$ with code smaller than 11, we add a_1 to $v.pow$ (lines 10–12). Hence, a node gains a higher power if it has more non (1,2)-dominated nodes in its neighbourhoods. By assigning different values to a_1, a_2, we may obtain different results.

Since there might be some links among nodes in $N_1(v) \cup N_2(v)$, the algorithm not only updates the power of v, but also nodes in $N_1(v) \cup N_2(v)$. For this reason the complexity of this algorithm is $O(\Delta^2)$, where Δ is the maximum degree of a node in a graph.

3.3. Algorithm `Minimal_12_Set`

The second algorithm, namely `Minimal_12_Set` uses the `PowerOfNodes` algorithm and finds a minimal (1,2)-dominating set.

Algorithm `Minimal_12_Set` Analysis

In the beginning, Algorithm 2 assigns initial values to nodes: *code* and *dom* are assigned 0, while *pow* of a node v is a positive number and is equal to $1 + a_1 \cdot |N_1(v)| + a_2 \cdot |N_2(v)|$ (lines 2–5). In each step of the main loop of the `Minimal_12_Set` (while…do lines 6–20), a node with the maximum value of *pow* is chosen and added to the minimal (1,2)-dominating set (lines 7–11). This node changes its value of *dom* from 0 to 1 (line 12). Then, necessary local variables of nodes are updated, first *code* (lines 14–19), then *pow* (line 20). Checking whether $x.pow > 0$ (line 11) prevents the addition of a redundant node to the minimal (1,2)-dominating set. At the end of the performance of the Algorithm 2 (line 21), nodes with *dom* equal to 1 form a minimal (1,2)-dominating set.

Note that in the case of assigning $x.dom = 1$ the powers of nodes at distance at most 4 might change. Our algorithm updates only powers of nodes at distance at most 2 from x (see the `Minimal_12_Set` algorithm). This accelerates the performance of the algorithm. However lines 7–11 make sure that the set of nodes with $dom = 1$ is always minimal in sense of (1,2)-domination.

Algorithm 2: `Minimal_12_Set`

Data: A simple graph G and parameters a_1 and a_2
Result: A minimal (1,2)-dominating set

```
1  Function Minimal_12_Set(G, a_1, a_2):
2      for v ∈ V(G) do
3          v.pow ← 1 + a_1 · |N_1(v)| + a_2 · |N_2(v)|;
4          v.code ← 0;
5          v.dom ← 0;
6      while Σ_{v∈V(G)} v.pow > 0 do
7          repeat
8              Let x be a node with the highest power;
9              update x.pow;
10         until x.pow > 0 or Σ_{v∈V(G)} v.pow = 0;
11         if x.pow > 0 then
12             x.dom ← 1;
13             x.code ← 30;
14             for y ∈ N_2(x) do
15                 if y.code ∈ {0, 10} then
16                     y.code ← y.code + 5;
17             for y ∈ N_1(x) do
18                 if y.code < 11 then
19                     y.code ← y.code + 10;
20             G ←PowerOfNodes(G, x, a_1, a_2);
21     return G
```

The main loop of the Algorithm 2 (while…do lines 6–20) performs at most $|V(G)|$ times, because each time it adds one node to the (1,2)-dominating set and nodes in the (1,2)-dominating set have $pow = 0$. Moreover, it takes at most $n = |V(G)|$ operations to choose a node with the highest power and at most Δ^2 operations to update the power of a node and nodes at distance 1 or 2 from it. Therefore, the time complexity for Algorithm 2 is $O(\Delta^2 n^2)$, so it is polynomial.

3.4. Example

We illustrate the performing of the `Minimal_12_Set` for $a_1 = 2$ and $a_2 = 1$. Let G be a graph as in Figure 2. The first number inside a node is its power, while the second–code. The numbers in Figure 2 show their values after performing the first five lines of the Algorithm 2.

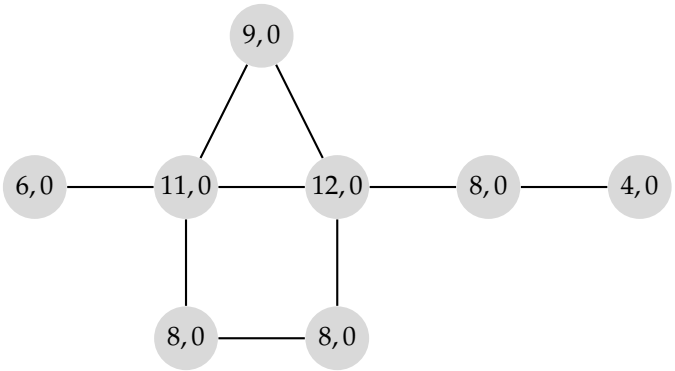

Figure 2. Graph G at line 5 of the Algorithm 2.

The graph in Figure 3 illustrates the state of the graph after performing the main while…do loop (lines 6–20) once. The dark node has $dom = 1$ and the light have $dom = 0$.

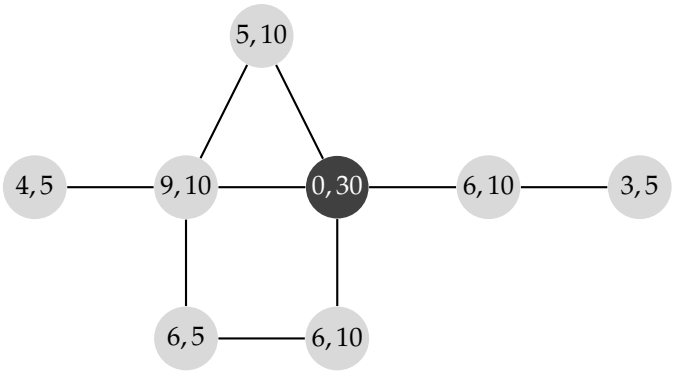

Figure 3. Graph G after performing the main loop of the Algorithm 2 once.

The state of the graph after performing the main loop twice is presented in Figure 4. Note that since the node of degree 1 and $pow = 3$ is at distance more than 2 from the node lastly added to the (1,2)-dominating set, its power is not updated.

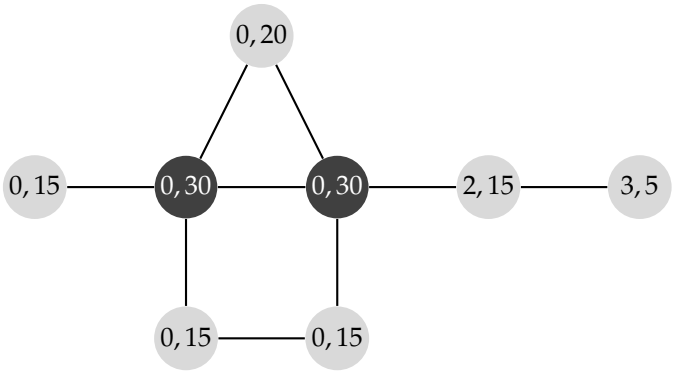

Figure 4. Graph G after performing the main loop of the Algorithm 2 twice.

The graph G after the last, third performance of the main loop is shown in Figure 5. The power of each node is 0. Since $\gamma_{(1,2)}(G) = 3$, in this example the `Minimal_12_Set` algorithm finds a minimal (1,2)-dominating set with the minimum possible cardinality.

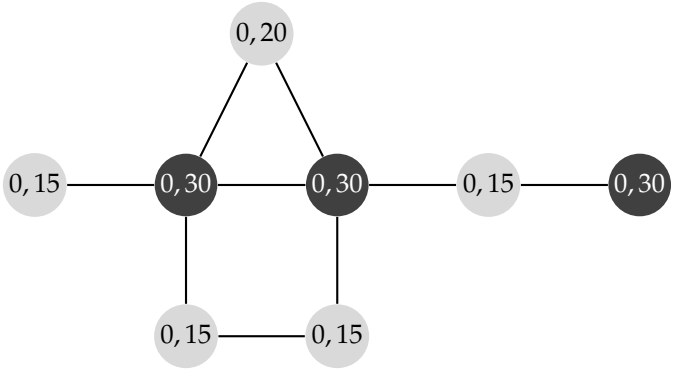

Figure 5. Graph G after performing the main loop of the Algorithm 2 third (and last) time.

4. Results

The `Minimal_12_Set` algorithm and `PowerOfNodes` function were implemented and tested using the *igraph* library in R. First, we used random geometric graphs as models of a network. A geometric random graph is the model of a spatial network, namely an undirected graph, constructed by placing randomly a given number of nodes in a unit square and connecting two nodes by a link if and only if their distance is in a given range, see [21,22]. In our tests, we generated random geometric graphs of order from 55 to

190 nodes and the radius within which the nodes are connected by a link between 0.08 and 0.4. We used three sets of parameters:

- $a_1 = 2, a_2 = 1$ denoted `2 1`,
- $a_1 = 1, a_2 = 1$ denoted `1 1`,
- $a_1 = 1, a_2 = 2$ denoted `1 2`.

For each graph we compared the results obtained by `Minimal_12_Set` with an algorithm that builds a minimal (1,2)-dominating set by choosing a permitted node randomly. (We permit a node to be added to a minimal (1,2)-dominating set if adding this node decreases the number of nodes which are not partially or fully (1,2)-dominated). We call this algorithm a random algorithm and denote it on presented diagrams by `rand`. Its time complexity is $O(\Delta^2 n)$.

Next, since there is an optimal algorithm finding the (1,2)-domination number in trees [18], we checked the performance of the algorithm for this class of graphs and we compared the results with an optimal solution as well as with the results returned by the random algorithm. Since adding a new link to a graph does not increase its (1,2)-domination number, the (1,2)-domination number of a spanning tree $T(G)$ of a connected graph G is an upper bound for the (1,2)-domination number of G. At last we shortly study the performance of our algorithm in random graphs and cubic graphs.

The statistics are done in Python using the pandas [23] library and the visualization with the seaborn [24] library.

4.1. Geometric Random Graphs

As it is shown in Figure 6, the `Minimal_12_Set` algorithm gave much better results than the random algorithm in the case of geometric random graphs. Not surprisingly, the difference between the results of the algorithms was greater for graphs with more nodes.

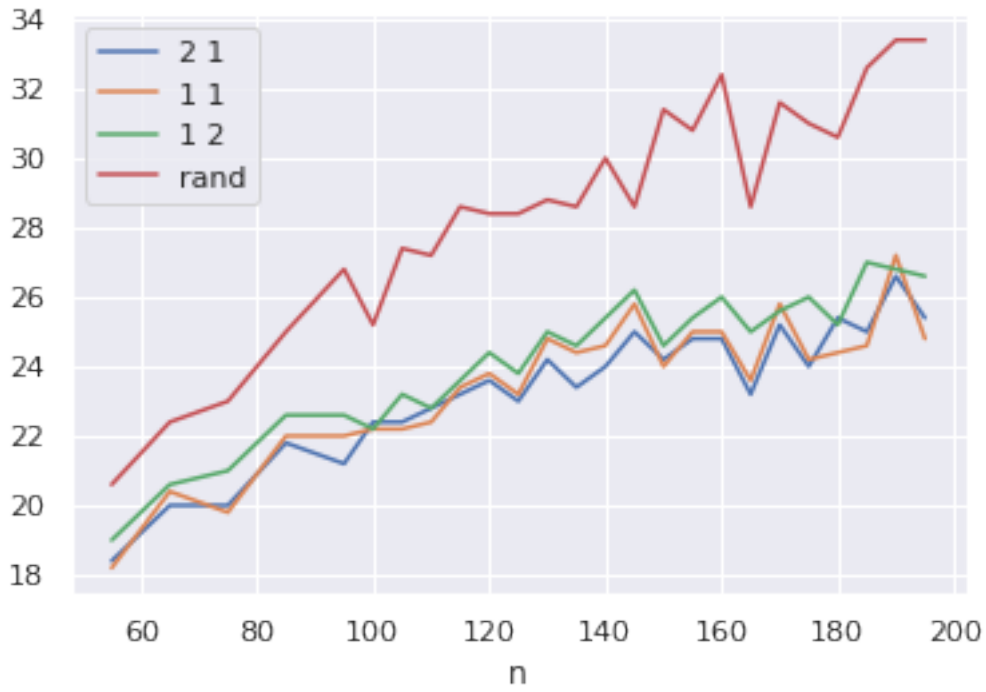

Figure 6. Average number of nodes in a minimal (1,2)-dominating set.

In most cases of our tests, the algorithm with parameters `2 1` gave the best results on average, however the differences between versions `2 1`, `1 1` and `1 2` were not significant.

4.2. Trees

The `Minimal_12_Set` algorithm analyzes local neighbourhoods of each node. For this reason, for example in stars it always finds a minimal (1,2)-dominating set with the smallest possible cardinality, namely a minimum (1,2)-dominating set. However, this may not be true for trees in general.

We tested the algorithms on random trees of order from 15 to 845 nodes. The results given by the `Minimal_12_Set` algorithm and the random algorithm were compared to the optimal algorithm that finds the (1,2)-domination number in trees (for details of the optimal algorithm see [18]). The cardinalities of the minimal (1,2)-dominating sets together with the optimal solutions are given in Figure 7.

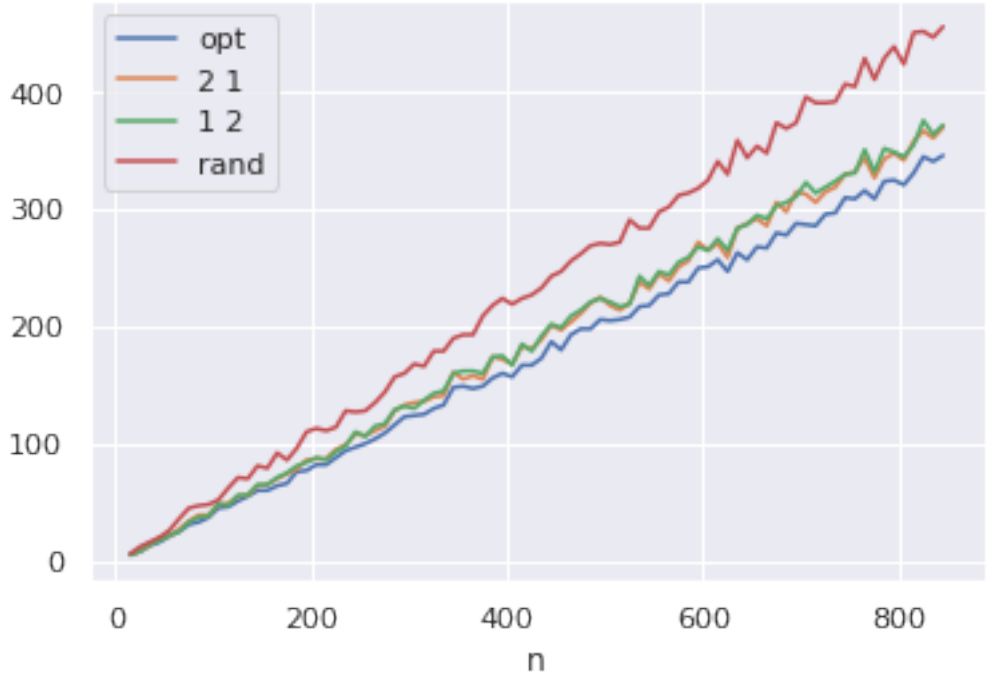

Figure 7. Returned values of carnality of minimal (1,2)-dominating sets for random trees.

It can be seen that the differences between the values returned by the `Minimal_12_Set` algorithm and the exact algorithm were much smaller than for the case of random algorithm. To investigate this difference more deeply, we counted the percent error. Namely, Figure 8 shows the percent error of the minimum value returned by algorithms `2 1` and `1 2` in relation to the optimal solution (denoted `% alg`), as well as the percent error of the random algorithm (denoted `% rand`). We noted that the `Minimal_12_Set` algorithm gives much better results than the random one.

The statistics presented in Table 1 show that the percent error of `2 1` and `1 2` is relatively small, because it lies between 0% and 12.12% with a mean value of 6.64%, while the percent error of the random algorithm is between 15.56% and 50.00% with a mean value of 33.23%.

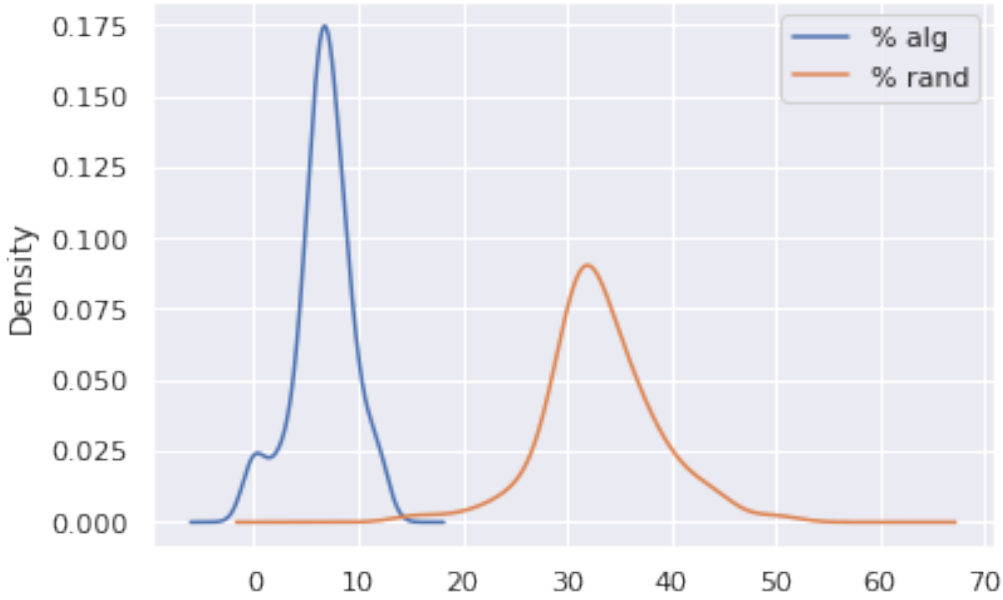

Figure 8. Percent error.

Table 1. Basic statistics for percent error.

	% Alg	% Rand
mean	6.636940	33.234861
std	2.700715	5.330012
min	0.000000	15.555556
25%	5.559414	30.740952
50%	6.698718	32.365320
75%	8.016610	35.875975
max	12.121212	50.000000

4.3. Random Graphs and Regular Graphs

We have tested the algorithm for random graphs generated according to the $G(n, p)$ Erdos–Renyi model for n varying from 30 to 250 and the probability for constructing a link $p = 0.1$. The results can be seen in Figure 9.

The statistics show that the random algorithm returned values on average 77.30% worse than the `Minimal_12_Set` algorithm. Moreover, the results given by the random algorithm were more diverse. In some cases the results were just 8.33% worse than the `Minimal_12_Set` algorithm, while in some cases were as high as 129.41% worse.

Since the `Minimal_12_Set` algorithm prefers the nodes with the highest degree, we tested the `Minimal_12_Set` algorithm and random algorithm on random cubic graphs, that is graphs in which each node is of degree 3. We supposed that these graphs should eliminate the advantage of adding nodes with high degrees to the minimal (1,2)-dominating set. Since the degree of each node is only 3, adding a node to a minimal (1,2)-dominating set changes the power of at most 10 nodes. We tested the algorithms for cubic graphs of order from 30 to 500 nodes. Furthermore, in this situation the `Minimal_12_Set` algorithm gave better results than the random one. The random algorithm gave worse results by 38.34% on average, varying from 18.42% to 75.00%.

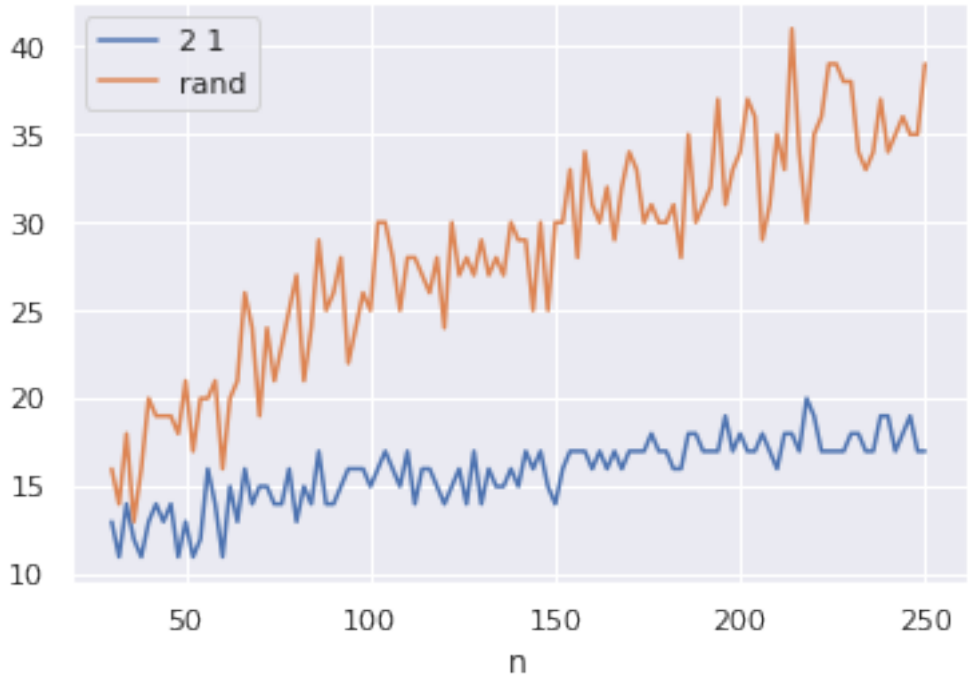

Figure 9. The cardinality of (1,2)-dominating sets returned by the `Minimal_12_Set` and random algorithms in random graphs.

5. Discussion

In this paper, we proposed two algorithms: `PowerOfNodes` and `Minimal_12_Set`. The second algorithm uses the first one and finds, in a polynomial time, a minimal (1,2)-dominating set of any graph. The conducted tests in trees show that the results returned by the `Minimal_12_Set` are not far from the optimal solution and are much better than a simple random algorithm. Since a tree is a subgraph of any graph, this allows us to assume that similar good results should be true in general graphs. The tests performed for geometric random graphs, random graphs and cubic graphs seem to confirm our suppositions. Even though the nodes in cubic graphs have the same degree, the new algorithm still returned better results than the random one.

For these reasons, our algorithm may be used in high reliability networks in which we must guarantee that each node has an instant access to a spare server at least at distance 2 from it. Since each node has a spare dominating node in (1,2)-dominating set, these networks are more fault tolerant. Additionally, if the model of a network is vast, there is a possibility of dividing it into unit squares and applying the algorithm in each square independently.

6. Conclusions and Future Work

In future, it might be interesting to investigate the potential of using the `Minimal_12_Set` algorithm in ad hoc networks in which nodes are changing their positions. Furthermore, it is worth checking whether updating the powers of all nodes at distance at most 4 from the node newly added to the minimal (1,2)-dominating set will give much better results than updating the powers of nodes at distance at most 2. Of course, the computational time complexity of the new version of the algorithm will be greater, so the question is if such greater time complexity is of practical importance.

Another interesting open problem is to investigate possible generalization of the presented algorithms to (1,k)-domination. In this paper, we are studying the case for $k = 2$, however other values of k seems to be worth studying. The possible extension of the algorithm for the neural networks and neuromorphic computing is left for further investigations in future.

Funding: This work was supported by the Ministry Subsidy for Research for Gdańsk University of Technology.

Data Availability Statement: Datasets analyzed or generated during the study. https://github.com/JoannaRaczek/Polynomial-Algorithm-for-Minimal-1-2-Dominating-Set, accessed on 18 December 2021.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Information System on Graph Classes and Their Inclusions (ISGCI). Available online: https://www.graphclasses.org (accessed on 18 December 2021).
2. Garey, M.R.; Johnson, D.S. *Computers and Intractability: A Guide to the Theory of NP-Completeness*; Freeman: San Francisco, CA, USA, 1979.
3. Haynes, T.W.; Hedetniemi, S.T.; Slater, P. *Fundamentals of Domination in Graphs*, 1st ed.; CRC Press: Boca Raton, FL, USA, 1998. [CrossRef]
4. Haynes, T.W.; Hedetniemi, S.T.; Slater, P.J. *Domination in Graphs Advanced Topics*, 1st ed.; Routledge: London, UK, 1998. [CrossRef]
5. Karbasi, A.H.; Atani, R.E. Application of Dominating Sets in Wireless Sensor Networks. *Int. J. Secur. Its Appl.* **2013**, *7*, 185–202.
6. Dubois, S.; Kaaouachi, M.H.; Petit, F. Enabling Minimal Dominating Set in Highly Dynamic Distributed Systems. In *Stabilization, Safety, and Security of Distributed Systems*; Pelc, A., Schwarzmann, A., Eds.; SSS 2015, Lecture Notes in Computer Science; Springer: Cham, Switzerland, 2015; Volume 9212._4. [CrossRef]
7. Yu, J.; Wang, N.; Wang, G.; Yu, D. Connected dominating sets in wireless ad hoc and sensor networks—A comprehensive survey. *Comput. Commun.* **2013**, *36*, 121–134. [CrossRef]
8. Yamuna, M.; Karthika, K. Medicine names as a DNA sequence using graph domination. *Pharm. Lett.* **2014**, *6*, 175–183.
9. Guze, S. An application of the selected graph theory domination concepts to transportation networks modelling. *Sci. J. Marit. Univ. Szczec.* **2017**, *52*, 97–102.
10. Ramalakshmi, R.; Radhakrishnan, S. Weighted dominating set based routing for ad hoc communications in emergency and rescue scenarios. *Wirel. Netw.* **2015**, *21*, 499–512. [CrossRef]
11. Hedar, A.-R.; Abdulaziz, S.N.; Mabrouk, E.; El-Sayed, G.A. Wireless Sensor Networks Fault-Tolerance Based on Graph Domination with Parallel Scatter Search. *Sensors* **2020**, *20*, 3509. [CrossRef] [PubMed]
12. Hedar, A.-R.; El-Sayed, G.A. Parallel genetic algorithm with elite and diverse cores for solving the minimum connected dominating set problem in wireless networks topology control. In Proceedings of the 2nd International Conference on Future Networks and Distributed Systems (ICFNDS '18), New York, NY, USA, 26–27 June 2018; Association for Computing Machinery: New York, NY, USA, 2018; pp. 1–9. [CrossRef]
13. Dai, F.; Wu, J. An extended localized algorithm for connected dominating set formation in ad hoc wireless networks. *IEEE Trans. Parallel Distrib. Syst.* **2004**, *15*, 908–920. [CrossRef]
14. Hedetniemi, S.M.; Hedetniemi, S.T.; Kinsley, J.; Rall, D.F. Secondary Domination in Graphs. *AKCE J. Graphs Combin.* **2008**, *5*, 103–115.
15. Factor, K.A.S.; Langley, L.J. An introduction to $(1,2)$-domination graphs. *Congr. Numer.* **2009**, *199*, 33–38.
16. Kayathri, K.; Vallirani, S. $(1,2)$-Domination in Graphs. In *Theoretical Computer Science and Discrete Mathematics*; Arumugam, S., Bagga, J., Beineke, L., Panda, B., Eds.; ICTCSDM 2016, Lecture Notes in Computer Science; Springer: Cham, Switzerland, 2017; Volume 10398. [CrossRef]
17. Michalski, A.; Włoch, I. On the existence and the number of independent (1,2)-dominating sets in the *G*-join of graphs. *Appl. Math. Comput.* **2020**, *377*, 125155. [CrossRef]
18. Raczek, J. Complexity Issues on Secondary Domination Number. *Nord. J. Comput.* **1994**, *1*, 157–171.
19. Kakugawa, H.; Masuzawa, T. A self-stabilizing minimal dominating set algorithm with safe convergence. In Proceedings of the 20th IEEE International Parallel and Distributed Processing Symposium, Rhodes, Greece, 25–29 April 2006; p. 8. [CrossRef]
20. van Rooij, J.M.M.; Bodlaender, H.L. Exact algorithms for dominating set. *Discr. Appl. Math.* **2011**, *159*, 2147–2164. [CrossRef]
21. Penrose, M. *Random Geometric Graphs*; Oxford Univ. Press: Oxford, UK, 2003. [CrossRef]
22. Bringmann, K.; Friedrich, T. Exact and Efficient Generation of Geometric Random Variates and Random Graphs. In *Automata, Languages, and Programming*; Fomin, F.V., Freivalds, R., Kwiatkowska, M., Peleg, D., Eds.; ICALP 2013, Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2013; Volume 7965. [CrossRef]
23. Pandas. Available online: https://pandas.pydata.org/ (accessed on 2 October 2021).
24. Seaborn: Statistical Data Visualization. Available online: https://seaborn.pydata.org/ (accessed on 2 October 2021).